# PATTERN RECOGNITION AND CLASSIFICATION USING ADAPTIVE LINEAR NEURON DEVICES

JOHN P. McGAHAN
MARSHALL J. TREADO

46

# PATTERN RECOGNITION AND CLASSIFICATION USING ADAPTIVE LINEAR NEURON DEVICES

[redacted]

by

John P. McGahan

Lieutenant, United States Navy

and

Marshall J. Treado

Major, United States Marine Corps

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
ELECTRICAL ENGINEERING

United States Naval Postgraduate School
Monterey, California

1 9 6 4

# PATTERN RECOGNITION AND CLASSIFICATION USING ADAPTIVE LINEAR NEURON DEVICES

by

John P. McGahan

and

Marshall J. Treado

This work is accepted as fulfilling

the thesis requirements for the degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

from the

United States Naval Postgraduate School

## ABSTRACT

Pattern recognition and classification systems have been under development for several years. This paper examines one of these systems, which has been called an adaptive linear neuron, to determine how the desired classification is achieved and how this system might be used in the practical field of character recognition. Specifically, the following ideas are discussed in this paper:

(1) The basic concepts of linear separability and iterative adaption by an adaptive linear neuron (Adaline), as applied to the pattern recognition and classification problem.

(2) Four possible iterative adaption schemes which may be used to train an Adaline.

(3) Use of Multiple Adalines (Madaline) and two logic layers to increase system capability.

(4) Use of Adaline in the practical fields of Speech Recognition, Weather Forecasting and Adaptive Control Systems and the possible use of Madaline in the Character Recognition field.

TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS

LIST OF TABLES

## 1. Introduction.

Systems which can be trained to classify complex digital and analog patterns have been under development for several years. One such system has been proposed by B. Widrow and others at Stanford University [1]. In this system, pattern recognition and classification are accomplished through the use of integrative memory cells, each of which consists of a variable resistor whose value can be made, by suitable "training", to become a useful function of the experiences of the device itself. The training of these devices is essentially a process of iterative adaption. In general, training consists of a systematic adjustment of each of the memory elements of the device in such a way that the system is forced to produce a predesignated output to each of a number of specific inputs. After the device has successfully adapted to a large number of training patterns, it has the ability to classify inputs which are related to the training patterns, as well as the training patterns themselves. Characteristics of the trained system are the effectively instantaneous classification of pattern inputs and the diffuse storage of information throughout the memory elements. The basic system developed by the Stanford group is an adaptive threshold device called an adaptive neuron, or Adaline, an acronym of <u>ada</u>ptive <u>li</u>near <u>ne</u>uron. A discussion of the functions performed by this element will be presented later. Systems containing one or more of these devices are being developed for use in speech recognition, weather forecasting and electrocardiogram analysis. They have also been used as trainable controllers in adaptive control systems.

In this paper it is planned to review the progress made by the Stanford group in the field of pattern recognition and classification.

Specifically, a close examination will be made of an Adaline, the basic ideas behind its use, the procedure followed in training it and the results obtained using a trained system. The investigation will be prosecuted experimentally by testing digital computer simulated models. In addition, a similar mathematical model will be applied to the practical problem of classifying the ten characters zero through nine when these are in a hand printed form. Finally, a start will be made on the problem of identifying a set of 45 alpha-numeric characters, also in hand printed form.

2. The Basic Adaptive Neuron.

The basic adaptive neuron, or Adaline, developed by the Stanford group is shown in figure 1. The input $p_i$'s are binary, but for convenience they are required to take the values of $\pm 1$, rather than the more conventional +1 and 0. An input pattern is defined to be a particular set of values of the $p_i$'s. In the Adaline each $p_i$ is multiplied by a corresponding (analog) weight, $w_i$, which can be regarded as the current setting of the $i^{th}$ memory cell, and which can take both positive and negative values. The weight $w_0$ is called the threshold weight. Thus the analog output of the summer is $\sum_{i=1}^{n} p_i w_i + w_0$ and the digital or quantized output is $\mathrm{sgn}\left\{ \sum_{i=1}^{n} p_i w_i + w_0 . \right\}$

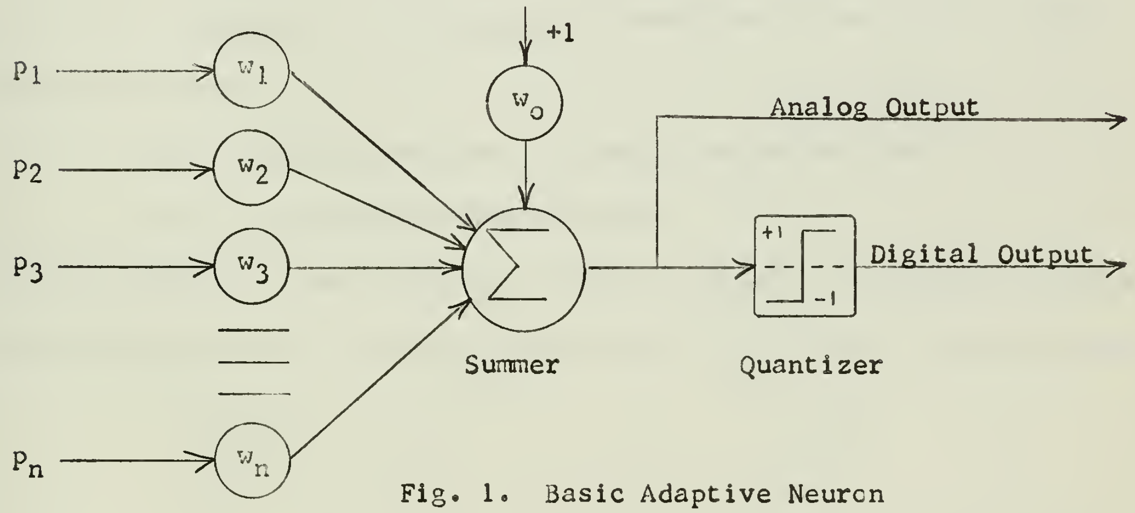


Fig. 1. Basic Adaptive Neuron

## 3. Linear Separability.

For simplicity, a two input Adaline will be discussed first. Suppose that it is desired to classify the possible input patterns into two classes:

(a) one or both $p_i$ positive, and

(b) both $p_i$ negative.

In this particularly simple case it is easy to see that if $w_0 = w_1 = w_2 = 1$, then the corresponding digital outputs will be +1 and −1. It will be said that this setting of the weights has classified the patterns into two classes, (a) with a +1 output, and (b) with a −1 output. In more complicated cases, the values of the weights would be adaptively determined during training. However, is such classification always possible? Suppose it is desired to classify the input patterns into the two classes:

(a) both $p_i$ positive, or both $p_i$ negative, and

(b) $p_i$'s with alternate signs.

In this case it is not possible to find a set of $w_i$ which will yield a digital output of +1 for the first class of input and −1 for the second.

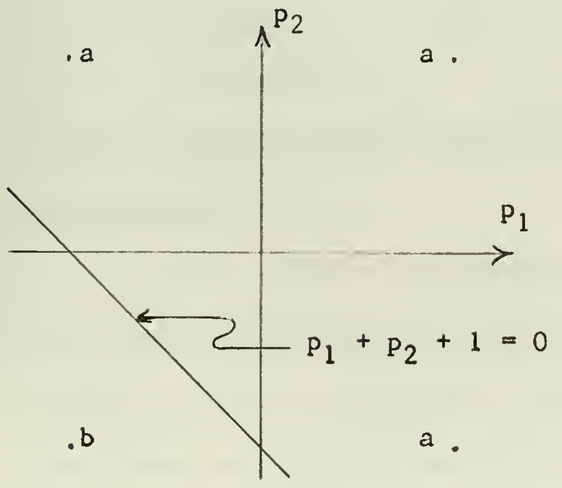


Fig. 2. Separation Into Two Output Classes

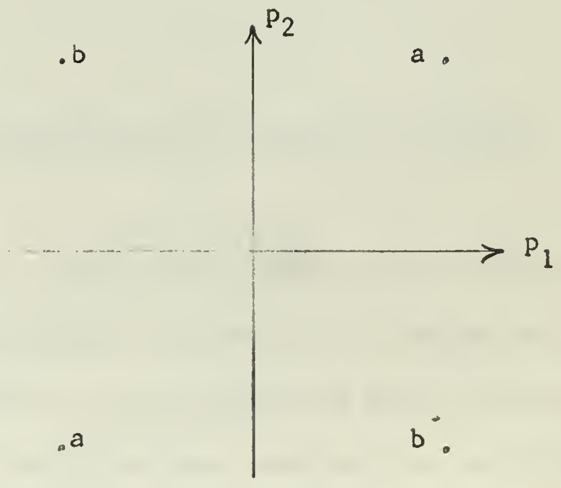


Fig. 3. Not Separable By One Adaline

It will then be said that the patterns are not "linearly separable" into these two classes and that a classification cannot be affected with a single Adaline.

The same thing can be illustrated graphically as in figures 2 and 3. The analog output must change sign as the line $\sum_{i=1}^{n} p_i w_i + w_0 = 0$ is crossed, and it can be seen that the analog output above the line in figure 2 is positive, and that it is negative below the line. Since a line can be drawn (actually an infinite number of lines can be drawn, each with different $w_i$) separating the a and b pattern classes, it can be concluded that the patterns are linearly separable into these two classes. On the other hand, it is immediately obvious that the classification of figure 3 is not possible.

It is of interest to continue the classification of the possible input patterns into different groups of two classes until all possible combinations have been examined. Using this procedure, both the number of linearly separable and the number of not linearly separable examples may be determined for a two input Adaline. The digital outputs for the two classes will be:

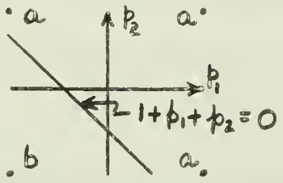
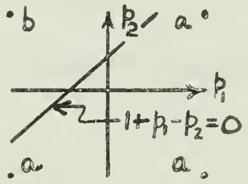
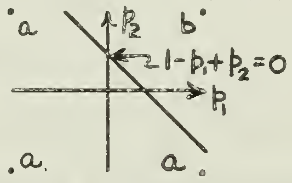
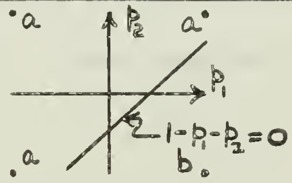
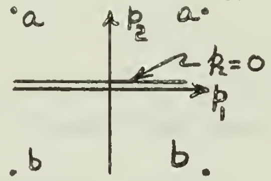
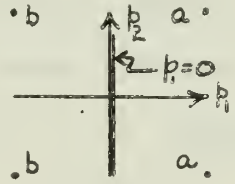
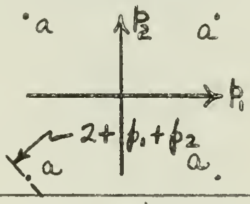
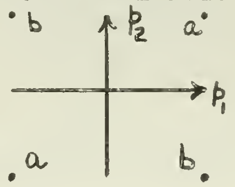
class (a) +1, and

class (b) -1.

Table 1 contains eight examples illustrating this concept of linear separability.

It can be seen that examples 1 thru 6 in the table could be repeated with the desired digital outputs reversed, i.e., with a -1 output for class (a) and a +1 output for class (b). This procedure would double the number of linearly separable classifications, and with the addition of the case illustrated as example 7 and its counterpart (for all inputs

Table 1. Separable and Not Separable Examples For Two Binary Inputs

| No. | Desired Classification of Input Patterns | Graphical Representation | One Set of $w_i$ For Linear Separation |
|---|---|---|---|
| 1 | (a) One or both $p_i$ + <br> (b) Both $p_i$ - |  | $w_0 = 1$ <br> $w_1 = 1$ <br> $w_2 = 1$ |
| 2 | (a) Both $p_i$'s + and both $p_i$'s -, $p_1$ + and $p_2$ - <br> (b) $p_1$ - and $p_2$ + |  | $w_0 = 1$ <br> $w_1 = 1$ <br> $w_2 = -1$ |
| 3 | (a) All except in (b) <br> (b) Both $p_i$'s + |  | $w_0 = 1$ <br> $w_1 = -1$ <br> $w_2 = -1$ |
| 4 | (a) Both $p_i$'s +, $p_i$'s -, $p_1$ - and $p_2$ + <br> (b) $p_1$ + and $p_2$ - |  | $w_0 = 1$ <br> $w_1 = -1$ <br> $w_2 = 1$ |
| 5 | (a) Both $p_i$'s +, $p_1$ - and $p_2$ + <br> (b) Both $p_i$'s -, $p_1$ + and $p_2$ - |  | $w_0 = 0$ <br> $w_1 = 0$ <br> $w_2 = -1$ |
| 6 | (a) Both $p_i$'s +, $p_1$ and $p_2$ - <br> (b) Both $p_i$'s -, $p_1$ - and $p_2$ + |  | $w_0 = 0$ <br> $w_1 = 1$ <br> $w_2 = 0$ |
| 7 | (a) All inputs + |  | $w_0 = 1$ <br> $w_1 = ½$ <br> $w_2 = ½$ |
| 8 | (a) Both $p_i$'s +, both $p_i$'s -, <br> (b) $p_i$'s w/ alternate signs |  | None |

to be classified as (b)), would yield a total of fourteen examples which are linearly separable. The interchange of the +1 and -1 of example 8 would yield a second example which is not linearly separable. Thus the total number of separable and unseparable classifications into two classes is equal to 16.

Suppose now that the number of binary inputs to an Adaline is 3 rather than 2, and that it is desired to classify the input patterns into the two classes:

(a) $p_1 = +1$, $p_2 = \pm 1$, $p_3 = \pm 1$, and

(b) $p_1 = -1$, $p_2 = \pm 1$, $p_3 = \pm 1$.

If the weights are chosen to be $w_1 = 1$ and $w_0 = w_2 = w_3 = 0$, it is easily seen that the digital output will be +1 for all the patterns in class (a), and -1 for those in class (b). Thus separation has been achieved by the plane $p_1 = 0$.

Table 2. Number of Linearly Separable Classifications for Different Numbers of Binary Inputs

| Number of Binary Inputs (n) | Maximum Possible Number of Examples ($2^{2^n}$) | Number of Known Linearly Separable Examples |
|---|---|---|
| 1 | 4 | 4 |
| 2 | 16 | 14 |
| 3 | 256 | 104 |
| 4 | 65,536 | 1,882 |
| 5 | $4.3 \times 10^9$ | 94,572 |
| 6 | $1.8 \times 10^{19}$ | 15,028,134 |

In general, if there are n binary inputs to an Adaline, the two classes will be separable if the $w_i$ can be chosen in such a way that the n-1 dimensional hyperplane $w_1p_1 + w_2p_2 + \cdots + w_np_n + w_0 = 0$ separates the two classes in n-space.

It was shown above that, for an Adaline with two binary inputs, the total number of input-output situations was 16. An extension of this reasoning from 2 to n binary inputs indicates that there are $2^{2^n}$ possible examples, again including both linearly and not linearly separable types. Table 2, which was extracted from [2], lists this figure together with the number of known linearly separable examples, for n from 1 to 6.

## 4. Threshold and Dead Zone.

Linearly separable input patterns cannot normally be separated unless the weights are carefully chosen. The process of changing the weights until each weight has the value most likely to affect the linear separation of the input patterns is called adaption or training.

One of the weights adjusted during adaption is $w_0$, the threshold weight. First, in relation to the binary or quantized output, we see that the output changes sign when $\sum_{i=i}^{n} p_i w_i = -w_0$ and the threshold can therefore be regarded simply as a bias on the quantizer, as shown in figure 4. Alternatively, if we consider the geometry of the hyperplane, it can be seen that a change of the threshold from $w_0$ to $w_0'$ affects a shift of the hyperplane to a new location parallel to its original position, as illustrated in figure 5 for n = 2. In fact, the perpendicular distance of the hyperplane from the origin, d, will be $\dfrac{w_0}{(w_1^2 + w_2^2 ---- w_n^2)^{\frac{1}{2}}}$. It should also be noted that the slope of the

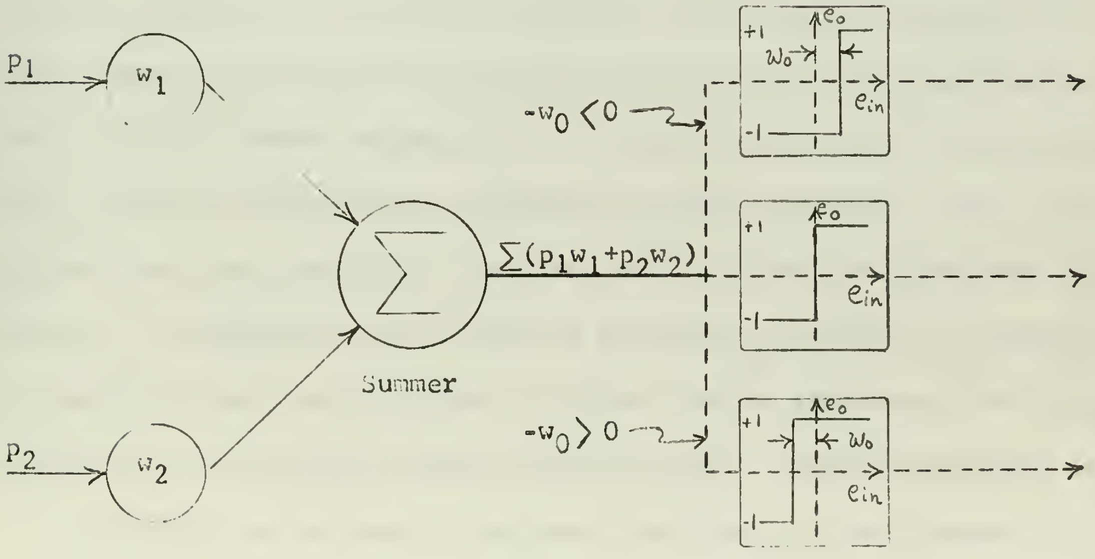


Fig. 4. Effect on Quantizer of Changes in Threshold Weight

hyperplane is not a function of the threshold, but instead is determined by the settings of the other weights.

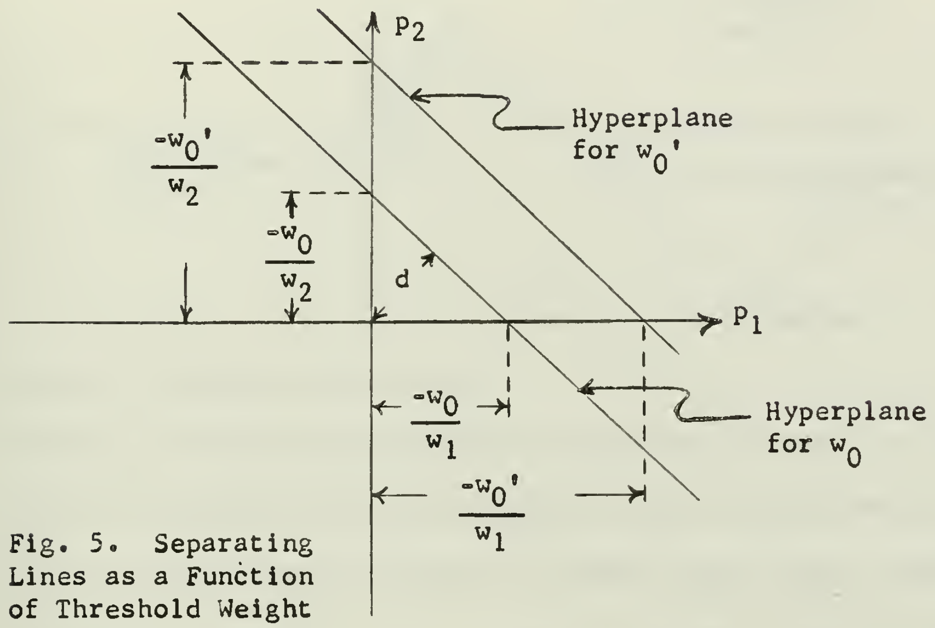


Fig. 5. Separating Lines as a Function of Threshold Weight

For the situation depicted in figure 6, any of the three hyperplanes determined by the thresholds $w_0$, $w_0'$ and $w_0''$ will linearly separate the input patterns into the two desired output classes a and b. However, use of the $w_0'$ or $w_0''$ hyperplanes would not provide the safety that might be needed in a practical situation. Some of the factors that might cause classification errors are resolution of the quantizer and drift in the circuit components. As a result, the input to the quantizer must always be sufficiently different from zero so that a small change in the quantizer input after adaption will not yield an incorrect digital output. In addition, the problem of resolution and drift is magnified by the fact that, as the number of binary inputs increases, the tolerance required on each weight becomes more critical. This is discussed in [3].

Now what can be done to decrease the effect of such errors on the Adaline system? One technique would be to utilize the $w_0$ hyperplane as the separating plane, with the $w_0'$ and $w_0''$ hyperplanes as the outer

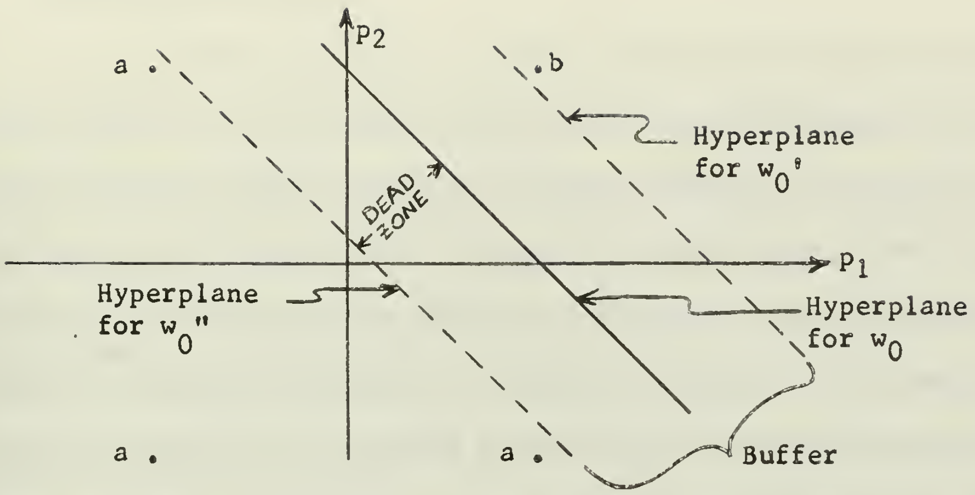


Fig. 6. Buffer and Dead Zone

limits of a buffer zone about the separating plane. The system would then be trained so that all possible input patterns would be classified into classes above and below the buffer zone, with a consequent reduction of the danger of producing an incorrect output from the quantizer. For convenience, each of the spaces on either side of the separating plane will be called a "dead zone" as labeled in figure 6.

## 5. Multilevel Adaline.

It is often necessary to separate the input patterns into more than two classes. In some cases this can be done effectively with a multi-level Adaline, which separates the input patterns into classes according to the analog output (level) assigned to each pattern. For this type of separation to be possible the input patterns or sets of input patterns must be linearly separable in a specific fashion. The several assigned output levels may be regarded as equivalent to a corresponding set of threshold weights. These in turn define a set of parallel hyperplanes such as shown in figure 6 for $w_0$, $w_0'$ and $w_0''$. Therefore, the use of the multilevel Adaline is restricted to the classification of inputs which can be linearly separated from each other by a series of parallel hyper-planes.

A typical use of a multilevel Adaline would be to classify the written digits one through nine. As a simple test of this concept a computer simulated Adaline was used to separate one sample of each of these digits. The written digits were converted into patterns using a seven by seven input space. After training, all nine patterns were correctly classified showing that this particular set of patterns was separable in this manner.

## 6. Adaline Adaption.

The previous sections contained a discussion of Adaline fundamentals and the concept of separating pattern classes by the appropriate setting of the weight elements. Training or adaption is the process of iteratively determining suitable values for the weight elements. This section will formulate the adaption problem and then consider four possible adaption or training procedures.

Adaline application in a pattern recognition problem involves the following sequence of events. Firstly, training patterns are chosen and assigned digital outputs, i.e. plus or minus one. Then an adaption process is selected and Adaline is trained until it either correctly identifies all the patterns, or until the training is terminated due to a failure of the training scheme to converge. If the training was not so terminated, the Adaline would now have the capability to correctly identify all the training patterns (thus proving the pattern classes to be separable) and in addition, the ability to recognize a number of "similar" patterns. In fact, one of Adaline's main advantages is the ability to identify patterns it has not been trained on. Other methods of pattern recognition such as "table look up" do not generally have this capability.

One question now to be answered is, what constitutes a "similar" pattern? One measure of the similarity of two patterns would be the number of pattern elements that would have to be changed in one pattern to make the two patterns identical. Figure 7 displays three patterns, each containing 16 pattern elements. Pattern (a) differs from pattern (b) by one pattern element, while pattern (a) differs from pattern (c) by eight pattern elements. Patterns (a) and (b) would be classified as "similar" patterns while patterns (a) and (c) would be classified as dissimilar patterns if the above criterion were used. In some cases, however, a pattern might be regarded as "similar" to a training pattern if it were related to it, for example, by a simple rotation or translation. In that case patterns (a) and (c) would be regarded as similar. This criterion might be used if the Adaline was being specifically trained to recognize patterns even when translated or rotated from their normal configuration.

```
X X X o      X X X o      o o o o

o X o o      o o o o      o X X X

o X o o      o X o o      o o X o

o o o o      o o o o      o o X o

  (a)          (b)          (c)
```

Fig. 7. Similar and Dissimilar Patterns

Finally, a "similar" pattern is often generated by the contamination of a training pattern by noise. For this reason "similar" patterns are sometimes referred to as "noisy" patterns. The ability of Adaline to recognize similar patterns is clearly an extremely complex function of the total number of pattern elements in the input pattern, the number and complexity of the training patterns, the value of the weight elements, and the degree of similarity of the input pattern to a training pattern.

It was shown earlier that the analog output is imposed on a quantizer whose digital output is either plus or minus one. It is usually desirable to endow the quantizer with a dead zone, so that some finite magnitude of the analog output is required before the digital output becomes non-zero. During the training or adaption phase the analog output of each training pattern is made equal to or greater than an "adaption dead zone". After training, a smaller value of dead zone may be introduced which will often enhance the probability of correctly identifying similar patterns. Thus patterns will be classified according to the relationship of their analog outputs to a "recognition dead zone".

The adaption or training problem can be formulated as:

Given: A collection of patterns and the desired digital output for each pattern.

Find: The value of the weight elements such that the expression $w_0 + \sum_{i=1}^{n} p_i w_i$ yields the appropriate analog output for each input pattern.

The weight values are usually determined by an iterative process based on a comparison of the actual analog output for each training pattern with the corresponding desired analog output. In other words, an input pattern is imposed on Adaline, the analog output is examined, the weight elements are changed if required, and then another pattern is imposed. This process is continued until the analog output of each pattern is acceptable without further adaption, or until it is determined that the process will not converge. The adaption process will be defined as having converged when each training pattern generates an acceptable output. It should be noted that an adaption process may not converge to a solution even though the pattern classes themselves are linearly separable. In such a case a different adaption technique would have to be employed.

7. Specific Adaption Schemes.

Common to each of the adaption schemes discussed below are the following rules.

When an analog output is unacceptable:

1. Change each of the weight elements by the same absolute magnitude.

2. Increase or decrease each weight according to whether the product of the corresponding pattern element by the desired change of the analog output is positive or negative. For example, if the analog output is to be increased and $p_1$ is $-1$, then $w_1$ would be decreased.

A description of the four adaption schemes listed in [4] now follows.

<u>Minimum Square Error Adaption</u>

This process adjusts the weight elements in such a manner that the analog output for each pattern is driven toward the same absolute magnitude, called the adaption dead zone. The error at each step is defined as the difference between the adaption dead zone and the actual analog output, and the change made to each weight element is expressed by the following equation:

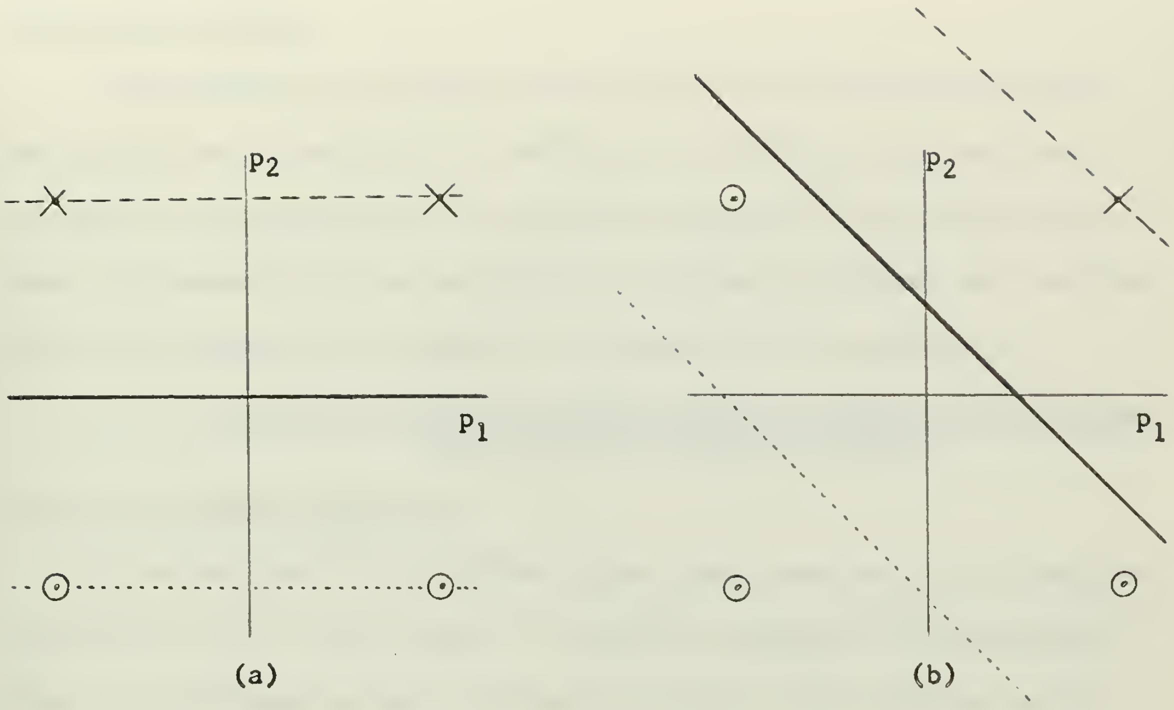
$$\left| \Delta w_i \right| = \frac{\text{Error x Proportional Constant}}{\text{Total Number of Weights}}$$

If the proportional constant is greater than zero and less than two, the process will converge provided that the patterns are both linearly separable and meet an additional requirement that will be discussed later. In the case where the weight elements are continuously variable, it has been experimentally determined (Appendix III), that the number of iterations required to converge in a typical situation approaches a minimum when the proportional constant is approximately equal to one.

The criterion requiring identical magnitudes for the analog outputs for all input patterns causes an extremely large number of iterations. Therefore, a tolerance is usually established around the adaption level, defined as minimum square error bound, and no changes are made to the weight elements if the analog output falls within this bound.

This adaption scheme also suffers from the disadvantage that there is no assurance of convergence, even if the pattern classes are known to be linearly separable. If the number of input patterns is equal to or greater than one plus the number of weight elements, there is a possibility that the adaption will not converge. The two pattern space, considered in section 3, will be used to illustrate this limitation. It can be seen in Figure 8 (a) that the classes can be separated in such a way that the analog output has the same magnitude for each pattern. This is not true in the case of the configuration of Figure 8 (b), and the minimum square error procedure, as described above, will not converge even though the classes are linearly separable.

The next three adaption procedures have been proved to converge to a solution provided that the pattern classes are linearly separable. These schemes compare the adaption dead zone value with the actual analog output and, if the magnitude of the analog output is less than the adaption dead zone value, some method will be employed to adjust the weights in such a way as to increase the analog output magnitude. However, if the analog output magnitude is equal to or greater than the adaption dead zone, no changes are made.



X +1 Class Patterns

⊙ -1 Class Patterns

- - - - - - $w_0 + \sum_{i=1}^{n} p_i w_i = +d$

———— $w_0 + \sum_{i=1}^{n} p_i w_i = 0$

············ $w_0 + \sum_{i=1}^{n} p_i w_i = -d$

Fig. 8. Separation With Minimum Square Error Procedure.

<u>Incremental Adaption</u>

When the analog output magnitude is less than the adaption dead zone value, all the weight elements are changed by an amount:

$$|\Delta w_i| = \frac{\text{Incremental Constant x Adaption Dead Zone}}{\text{Total Number of Weight Elements}}$$

The number of iterations required to converge, and the final value of the weight elements, is a function of the incremental constant. Note that the corrections are independent of the errors.

## Relaxation Adaption

This procedure is similar to the minimum square error adaption except that no corrections are made when the analog output magnitude is equal to or greater than the adaption dead zone. The adaption process will converge if the proportional constant is between zero and two. The weight elements are changed, if required, by an amount:

$$\left|\Delta w_i\right| = \frac{(\text{Adaption Dead Zone} - \text{Analog Output}) \times \text{Proportional Constant}}{\text{Total Number of Weight Elements}}$$

## Modified Relaxation Adaption

A disadvantage of the minimum square error and relaxation adaption procedures is the large number of iterations required for convergence. When the difference between the desired analog output and the actual analog output is small, a small correction is made. Thus, the closer the process gets to the solution, the smaller the magnitude of the corrective changes. The modified relaxation adaption procedure overcomes this difficulty by correcting to a value larger than the adaption dead zone. This value, usually 1.1 to 1.5 times the adaption dead zone magnitude, is defined as the adaption level. No corrections are made if the output magnitude exceeds the adaption dead zone. The equation for the change of weight elements, where the proportional constant should again be between zero and two for convergance, is:

$$\left|\Delta w_i\right| = \frac{(\text{Adaption Level} - \text{Analog Output}) \times \text{Proportional Constant}}{\text{Total Number of Weight Elements}}$$

## 8. Evaluation of Adaption Procedures.

The different characteristics of the adaption procedures were investigated by the solution of some typical problems. In all cases, Adaline was simulated on a CDC 1604 Digital computer using one of the adaption procedures which are defined by the Fortran statements listed in Appendix I. In one example the adaption process is examined after each iteration, (Appendix II), while in another a test is described of Adaline's ability to correctly identify a number of patterns upon which it had not been trained (Appendix IV).

If training patterns are not separable, the training process will not converge and must be arbitrarily terminated. The Adaline will then fail to identify all of the training patterns correctly, but if the number of failures is small this may be tolerable in some applications. When it comes to the recognition of noisy versions of the training patterns, it must be expected that Adaline will only recognize a statistical percentage of the similar patterns presented. The only method of ensuring that Adaline will correctly identify all possible input patterns is to train on all the conceivable patterns. But, this is then a form of "table look up" recognition, which can be performed by other means without the necessity of employing an iterative scheme. Appendix V summarized the results of training Adaline on pattern classes that are not separable.

No attempt will be made to promote the use of one adaptive procedure in lieu of the others. It can be noted that the modified relaxation procedure usually requires the fewest number of iterations to converge, but results in a wide spread in the analog output values. On the other hand, the minimum square error adaption requires more iterations to converge but has a narrow spread in the analog output values.

## 9. Possible Adaline Applications.

The output of a trained Adaline can be regarded as a binary digit, or logical decision, whose value depends upon the pattern input to the device. It follows, therefore, that an Adaline can, in principle, be applied in any situation where a decision is to be made on the basis of some "input" to the decision making device. In particular, the Adaline concept is valuable in situations where performance can or must be improved as experience accumulates.

The basic difficulties in the application of Adalines relate firstly to the problem of converting an input into a pattern, and secondly to the development of a suitable training procedure which will ensure that the Adaline does in fact improve its performance with practice. The following three sections will consider a few of the possible Adaline applications.

10. Servo-Mechanism Controller.

A single Adaline with its digital output can be used as a bang-bang servo-mechanism controller as in figure 9.

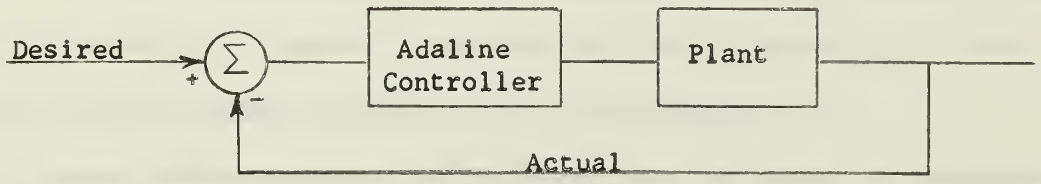


Fig. 9. Basic Adaline Servo-Mechanism Controller

Graduate students at Stanford University have used an Adaline in such a control problem. The plant consisted of a rolling cart powered by a reversible electric motor. Installed on the cart was an inverted pendulum. The Adaline controller was trained to keep the pendulum in a vertical position without extreme excursion of the cart in either direction. Four plant variables were measured; the position and velocity of the cart and of the pendulum.

The direction the electric motor should rotate, and thus the desired Adaline digital output, is a function of the four measured variables. The value of each variable can be cataloged into one of several distinct levels, and each level can in turn be represented by a code consisting of a series of pattern elements. These pattern codes must be carefully chosen to ensure that the pattern classes are linearly separable. The complete input pattern is composed of the pattern elements of the four variables.

The Adaline is trained by permitting it to observe the performance of another type of controller. The "correct" response is then available at all times and the Adaline weights can be adjusted to bring the Adaline output into agreement. After the training is completed, the Adaline can take over the operation of the plant.

## 11. Speech Recognition.

A real time speech recognition system [5] has been constructed at Stanford University. The system, which consists of several parallel Adalines, has the capability of converting speech into typewritten words. Since the operation of parallel Adalines will be discussed in a later section, only the coding technique will be discussed here.

The main problem encountered in coding was the choice of parameters to describe the sound of a spoken word. Bandpass filters were employed to separate the sound energy into eight frequency bands and the sound intensity in each band was then digitally coded according to the amplitude level. Four levels were chosen corresponding to the three bit patterns, 000, 001, 011, or 111, where 1 equals +1 and 0 equals -1.

Ten samples were taken during the utterance of a word, so that each filter generated 30 pattern elements. The complete pattern from all eight filters therefore consisted of 240 pattern elements.

12. Weather Forecasting.

Adalines have been applied in the area of weather forecasting to the extent of predicting "fair" or "rain" in one locality [6]. In this application parallel Adalines were trained to interpret weather maps. In particular, they were trained to "read" surface pressure maps of a 500,000 square mile area.

The weather map was divided into 48 regions each of approximately 600 square miles. Then, the expected range between the highest and lowest pressures was divided into ten levels, each of which was represented by one of the ten digits, 0 through 9. Thus, each input pattern contained 48 pattern elements each of which could acquire one of ten values (as compared with the usual two).

The results obtained from the Adalines were comparable with those obtained from "human" weather forecasters.

## 13. Classification When Pattern Classes Are Not Linearly Separable.

Previous sections have discussed the basic structure of an Adaline and have detailed several schemes for the use of an Adaline in pattern recognition and classification. The discussion thus far has been limited to the separation of sets of input patterns into classes by a single Adaline. It was found that the classification of input patterns into two classes could not always be affected with a single Adaline. This situation was illustrated by the not linearly separable example shown in figure 3. In that example it was desired to classify all input patterns into the two classes:

(a) both $p_i$ positive, or both $p_i$ negative, and

(b) $p_i$'s with alternate signs

This separation can be accomplished by a system using two Adalines in parallel. The weights of the two Adalines define two hyperplanes, which affect the desired separation as illustrated in figure 10. The overall system consists of two logic layers, the first layer being composed of Adalines with adaptive elements and the second layer made up of a fixed logic element or threshold device. In the first layer each Adaline

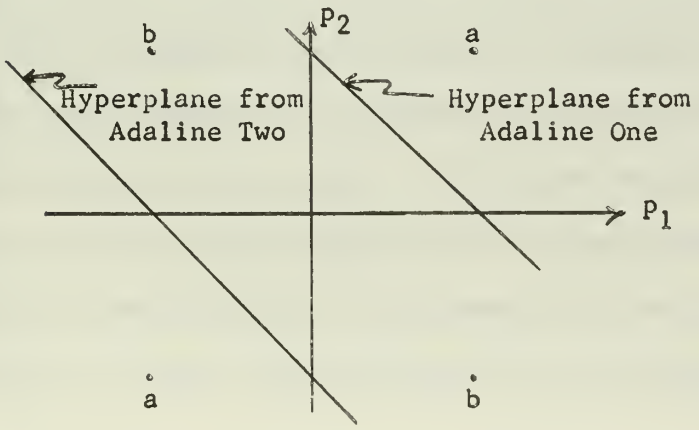


Fig. 10. Linear Separability With Multiple Adalines

attempts a classification of input patterns into linearly separable classes while the second layer combines the outputs of the first layer to complete the desired classification. Using the example of figure 10, the inputs would be classified in the first logic layer as follows:

| | |
|---|---|
| Adaline One | a) both $p_i$ positive |
| | b) all others |
| Adaline Two | a) both $p_i$ negative |
| | b) all others |

This classification would place the two hyperplanes as in figure 10. The next step would be to insert the Adaline outputs into the second layer which is an "or" gate[1] (in this case) in order to realize the desired output classification. This process is summarized in table 3.

Table 3. Separation Using Two Logic Layers

| Inputs | | Outputs | | | |
|---|---|---|---|---|---|
| | | Adaline #1 | Adaline #2 | Or Gate | Desired |
| 1 | 1 | 1 | -1 | 1 | 1 |
| 1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | 1 | 1 | 1 |

It is apparent that the choice of weights (hyperplanes) in the first level Adalines is dependent upon the logic device used in the second layer and vice versa. It follows that the choice of a logic device and the establishment of a training procedure for the Adalines may prove very difficult if the result is not known in advance, as it was here. In fact this would seem to be the major difficulty in this

[1]An "or" gate is a device which gives a positive output if one or more of its n inputs are positive.

scheme. One approach to this problem now follows.

There are many devices which could be used in the second layer to combine the Adaline outputs, so that it is important to estimate which of these devices is likely to be the best one for this purpose. The previously mentioned "or" element can be regarded as a special quantizer whose output is -1 unless at least <u>one</u> Adaline output is positive. At the other extreme would be an element whose output is -1 unless <u>all</u> n Adaline outputs are positive.

In trying to find the "best" quantizing device for use in the second logic layer it seems natural to try a device whose output becomes +1 when about $\frac{n}{2}$ of the Adaline outputs become positive. It has been found, [7], that for a system with an odd number of Adalines, a simple output majority, or a second layer "threshold" of $\frac{n+1}{2}$, will realize the classification of the greatest number of inputs. Similarly for an even number of Adalines, second layer thresholds of $\frac{n}{2}$ or $\frac{n+2}{2}$ will realize the highest percentage of classifications. It should be noted that the criterion used in the above reference to determine the "best" second layer threshold for general use was the criterion of the classification of the maximum number of input pattern sets. However, for any specific patterns, or sets of patterns, the threshold which is "best" might be anywhere from 1 to n.

There is still the problem of choosing the "best" adaption scheme for the first layer. One method that has been suggested makes use of both the analog output data from each Adaline and the digital output desired from the overall system. In the case where the second layer element is an "or" gate, the procedure to be followed will depend on the desired system output. Thus, if the desired system output is -1, all Adalines must give a negative output and any Adaline which is giving a positive output will have to be adapted. However, when all first layer outputs are negative and the desired system output is +1, only one Adaline need by adapted. In this case it has been suggested that adaption be confined to the Adaline whose analog output needs the smallest change to establish the required condition. If the second layer threshold is a majority logic device, the same general procedure will be followed. If, to obtain the desired system output, the output of at least k Adalines must be revised, the k Adalines whose outputs require the least change will be adapted. The idea behind this procedure is that adaption should take place with the minimum of disturbance to the previously established pattern of weights.

It may be noted that if it were not for the difficulty in choosing a suitable logic for the second layer, and a training procedure for the first, it would be theoretically possible to establish any desired classification if sufficient Adalines were employed in the first layer. In an extreme case, for example, the n+1 hyperplanes defined by the weights of n+1 Adalines could separate one input from all others in n-space _if_ the weights could be chosen properly, and _if_ the second logic layer properly interpreted the outputs of the Adalines.

## 14. Madaline.

Multiple Adalines in parallel, or a Madaline as this combination is sometimes called, may also be used to classify a set of input patterns into more than two output classes. If the input patterns can be appropriately separated by each of the Adalines, Madaline can accomplish the desired classification with the help of a digital output coding scheme such as that illustrated in figure 11 and table 4. It is easily seen that the maximum number of individual output codes available is $2^m$ where m is the number of Adalines in the Madaline. As m = 3 in figure 11 and table 4, it is readily apparent that eight input patterns or pattern sets can be classified into eight different digital output combinations in this situation.

Here the coding scheme is predetermined, so that the training follows the procedure devised for a single Adaline. That is, each Adaline is trained individually to generate the appropriate response to each of the training patterns.

The problem which may arise is related to the choice of codes for the several pattern sets. If these are not properly chosen, then the

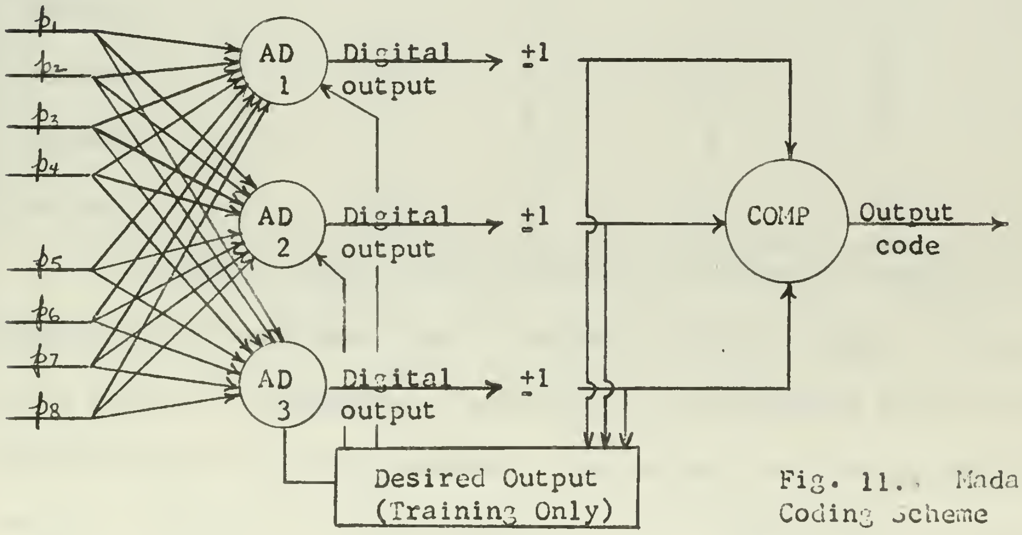


Fig. 11. Madaline Output Coding Scheme

individual Adalines may well be attempting to separate the patterns into classes which are not linearly separable. A change in the codes may eliminate the problem in any given situation, but no systematic procedure for choosing the coding scheme has been found.

Also to be considered is the fact that the coding scheme may affect the number of Adalines needed, and/or the number of adaptions necessary during training. These considerations, however, seem to be less important than that of the previous paragraph.

For the example of figure 11 all eight possible binary output codes are used. Note, however, that there are many possible choices for the code to be assigned to each pattern or pattern set. One such choice is shown in table 4.

Table 4. Typical Madaline Output Coding Scheme

| | Adaline 1 Output | Adaline 2 Output | Adaline 3 Output |
|---|---|---|---|
| pattern 1 | 1 | -1 | 1 |
| pattern 2 | -1 | 1 | -1 |
| pattern 3 | 1 | -1 | -1 |
| pattern 4 | -1 | 1 | 1 |
| pattern 5 | 1 | 1 | -1 |
| pattern 6 | -1 | -1 | 1 |
| pattern 7 | 1 | 1 | 1 |
| pattern 8 | -1 | -1 | -1 |

To summarize, each of the Adalines is trained to separate the input patterns into two classes (a) +1 output and (b) -1 output. Then, as an input pattern is imposed on the Madaline, the Adalines simultaneously determine their output responses. The outputs can then be examined to properly classify the imposed pattern. The classification of the written digits, one through eight, suggested herein was successfully accomplished using three computer-simulated seven by seven Adalines.

## 15. Character Recognition

In this section the application of a Madaline to the recognition and classification of hand printed letters, numbers and other punctuation symbols will be discussed. For definiteness, the alpha-numeric character set chosen corresponded to that used in Fortran computer coding, and the aim was to evaluate the possibility of "reading" hand printed Fortran symbols using a Madaline.

The investigation was divided into two parts. The first consisted of an evaluation of the feasibility of classifying the ten numeric characters using a Madaline, while the second consisted of an attempt to classify 45 Fortran characters using a similar device.

In the first part of the investigation the Madaline consisted of four Adalines, the minimum number required, each with a seven by seven input space. The first tests were conducted using patterns of ten digits obtained from five different persons, each of whom wrote one sample of each of the ten digits on a standard Fortran coding form. The written digits were enlarged by projection on a screen and a seven by seven grid form was used to determine the actual input patterns used. Each test digit was centered on the seven by seven grid form and a digit was defined to be "in" a grid space if it entered in such a manner that both sides of the line could be seen inside that space. The tests, which are detailed in Appendix VI, were conducted using the patterns obtained in this manner. In one of these tests, the Madaline was trained four times on each of the fifty different input patterns and was then asked to classify each input pattern. All input patterns were classified correctly.

The second part of this investigation was again conducted with the minimum number of Adalines required. Six Adalines, each with a seven by seven input space, were used to classify the 45 Fortran symbols. The results of this classification are detailed in Appendix VII.

## 16. Summary and Acknowledgements.

This paper contains a survey of the pattern recognition problem as approached through the application of adaptive linear neuron devices. Some of the basic Adaline concepts were verified experimentally and some concepts were amplified. In particular, an example was shown of the inability of the minimum square error adaption procedure to separate input patterns that were linearly separable, and a method was developed to display the convergence characteristics of the individual adaption procedures. In addition, preliminary tests to determine the feasibility of utilizing a Madaline for character recognition indicated that a possible application exists in this field.

An interesting project, which is a natural follow up of the work detailed herein, would be that of using an actual laboratory set up to more completely determine the feasibility of utilizing Madaline for character recognition. Perhaps this could be accomplished using photocells for input pattern detection. Another carry over project using Adalines would be to continue the weather forecasting analysis previously discussed [6]. The wealth of weather data available at this location makes this a particularly feasible project.

The authors wish to acknowledge the guidance and assistance given to them by Dr. J. R. Ward.

## BIBLIOGRAPHY

1. Widrow, B. Generalization and Information Storage in Networks of Adaline Neurons. 1962

2. Winder, R. O. Single Stage Threshold Logic. AIEE Fall Meeting 1960, 9 August 1960.

3. Mays, C. H. Solid State Electronics Research. Quarterly Status Report No. 10, Stanford Electronic Laboratories, 1 Jan. to 31 March, 1961.

4. Mays, C. H. Adaptive Threshold Logic. Report SEL-63-027, Stanford Electronic Laboratories, April, 1963.

5. Talbert, L. R. et al. A Real-Time Adaptive Speech-Recognition System. Report SEL-63-064, Stanford Electronics Laboratories, May, 1963.

6. Hu, M. J. A Trainable Weather-Forecasting System. Report SEL-63-055, Stanford Electronics Laboratories, June, 1963.

7. Ridgeway, W. C. III. An Adaptive Logic System With Generalizing Properties. Report SEL-62-040, Stanford Electronics Laboratories, April 1962.

## APPENDIX I

## FORTRAN STATEMENTS OF ADAPTION PROCEDURES

The Fortran statements used to simulate the four adaption procedures are listed below. The program was written for an Adaline having 17 weight elements. It is to be noted that this is not the complete program but only the adaption techniques.

Abbreviations used in the program:

| | |
|---|---|
| ADAPT | Adaption level |
| BETA | Increment constant |
| BMSE | Bound for minimum square error |
| DELTA | Dead zone level |
| PAT(I,J) | Input pattern, I=pattern element, J=input pattern number |
| PIE | Proportional constant |
| SGN(J) | Desired binary output of pattern J |
| SUM | Analog output of pattern J |
| WEY(I) | Value of weight element I |

```
C     MINIMUM SQUARE ERROR ADAPTION
      ERROR=SGN(J)*DELTA-SUM
      ABER=ABSF(ERROR)
      IF(ABER-BMSE) 100,100,50
   50 ENORM=PIE*(ERROR/17.0)
      DO 60  I= 1,17
   60 WEY(I)=WEY(I)+ENORM*PAT(I,J)
  100 CONTINUE
```

```
C     INCREMENTAL ADAPTION
      IF(SGN(J))  256,254,254
  254 IF(SUM)  258,258,255
  255 IF(SUM-DELTA)  258,262,262
  256 IF(SUM)  257,259,259
  257 IF(SUM+DELTA)  262,262,259
  258 SIGN=+1.0
      GO TO 260
  259 SIGN=-1.0
  260 DO 261  I=1,17
  261 WEY(I)=WEY(I)+SIGN*BETA*PAT(I,J)
  262 CONTINUE

C     RELAXATION ADAPTION
      IF(SGN(J))  306,304,304
  304 IF(SUM)  310,310,305
  305 IF(SUM+.0005-DELTA)  310,312,312
  306 IF(SUM)  307,310,310
  307 IF(SUM-.0005+DELTA)  312,312,310
  310 ERROR=SGN(J)*DELTA-SUM
      ENORM=PIE*(ERROR/17.0)
      DO 311  I=1,17
  311 WEY(I)=WEY(I)+ENORM*PAT(I,J)
  312 CONTINUE
```

```
C     MODIFIED RELAXATION ADAPTION
      IF(SGN(J))  356,354,354
  354 IF(SUM)  360,360,355
  355 IF(SUM-DELTA)  360,362,362
  356 IF(SUM)  357,360,360
  357 IF(SUM+DELTA)  362,362,360
  360 ERROR=SGN(J)*ADAPT-SUM
      ENORM=PIE*(ERROR/17.0)
      DO 361  I=1,17
  361 WEY(I)=WEY(I)+ENORM*PAT(I,J)
  362 CONTINUE
```

# APPENDIX II

## EXAMPLES OF ADAPTION PROCEDURES

An example of pattern separation was solved using each of the four different adaption procedures to illustrate their characteristics. The test patterns were:

```
        X . X .     . X . X     . . . .     . . . .
(+1)    . X . .     . . X .     X . X .     . X . X
        X . X .     . X . X     . X . .     . . X .
        . . . .     . . . .     X . X .     . X . X
          (1)         (2)         (3)         (4)

        . . X .     . . . X     . . . .     . . . .
(-1)    X . X .     . X . X     . . X .     . . . X
        X X X .     . X X X     X . X .     . X . X
        . . . .     . . . .     X X X .     . X X X
          (5)         (6)         (7)         (8)
```

When a pattern is imposed on an Adaline, corrections are made to the weights so that its analog output satisfies the training criteria. This, however, has a tendency to partially destroy some of the effects of previous training. In an attempt to illustrate this process, the analog output of the Adaline was examined after each adaption throughout this test.

The fixed conditions of this experiment were:

Weights: Continuously variable with initial values of 0.0

Pattern sequence: 1, 2, 3, 4, 5, 6, 7, 8

Proportional constant: 1.00

Minimum square error adaption level: 30.00

Minimum square error adaption bound: $\pm$ 1.00

Incremental constant: 1.00

Adaption dead zone: 30.00

Adaption level: 40.00

Minimum Square Error Adaption

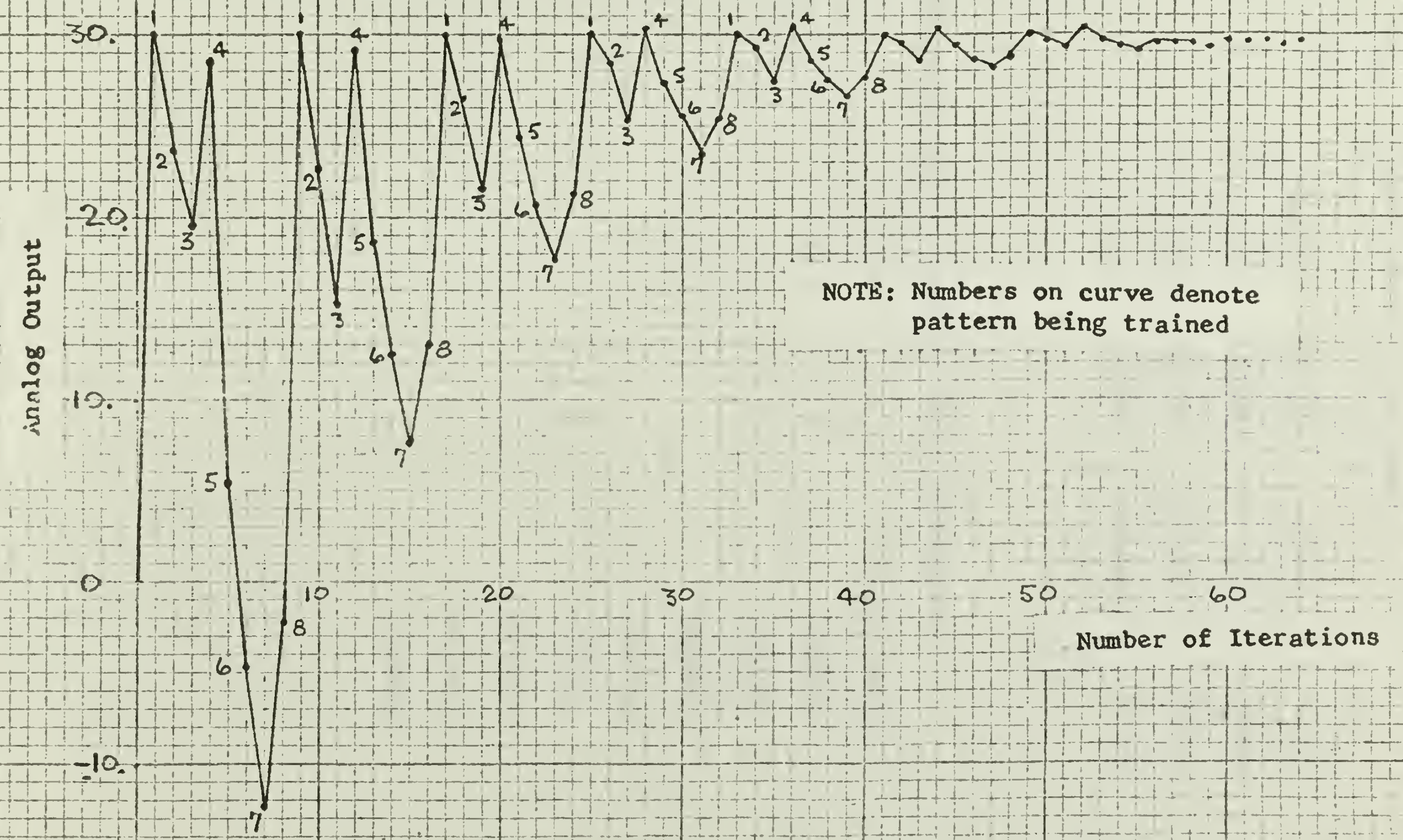
Number of iterations required to converge: 64

Weight elements after convergence.

```
3.436
             7.864     10.197     -5.141     -5.773

             -.725      9.090       .228    -11.528

           -17.821    -12.380    -16.595    -15.091

             -.400     -8.237     -9.718      4.442
```

Analog output of each pattern after convergence.

| Pattern | Analog output |
|---|---|
| 1 | 29.819 |
| 2 | 29.385 |
| 3 | 29.031 |
| 4 | 29.368 |
| 5 | -29.846 |
| 6 | -29.529 |
| 7 | -30.063 |
| 8 | -30.000 |



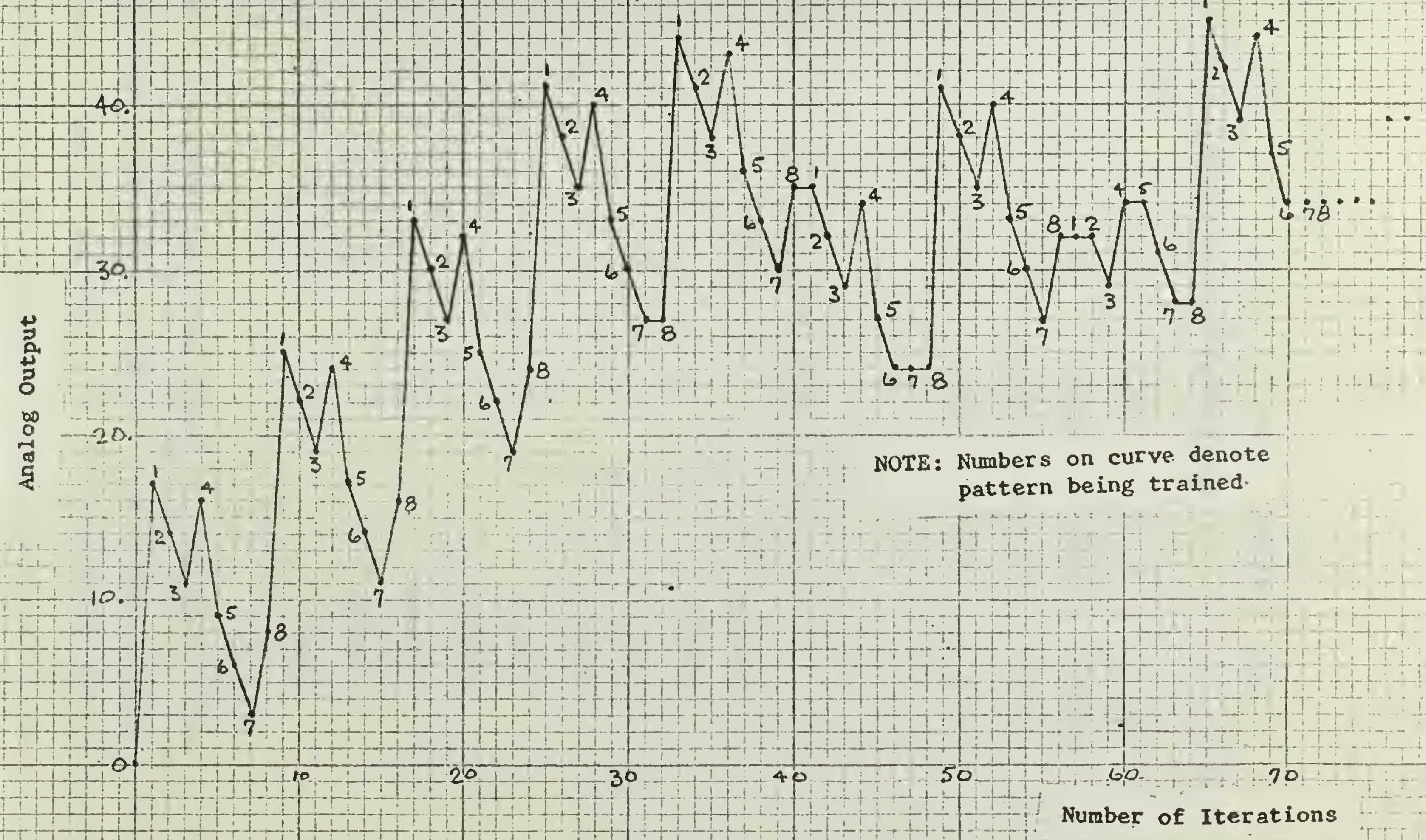
MINIMUM SQUARE ERROR ADAPTION

Incremental Adaption

```
    Number of iterations required to converge:  82

    Weight elements after convergence.

    4.000
                 10.000      14.000      -6.000       -6.000

                 -2.000      10.000        .000      -14.000

                -22.000     -14.000     -22.000      -16.000

                 -2.000     -10.000     -12.000        6.000
```

Analog output of each pattern after convergence.

| Pattern | Analog output |
|---|---|
| 1 | 30.000 |
| 2 | 46.000 |
| 3 | 30.000 |
| 4 | 30.000 |
| 5 | -42.000 |
| 6 | -34.000 |
| 7 | -46.000 |
| 8 | -30.000 |



INCREMENTAL ADAPTION

Relaxation Adaption

Number of iterations required to converge: 168

Weight elements after convergence.

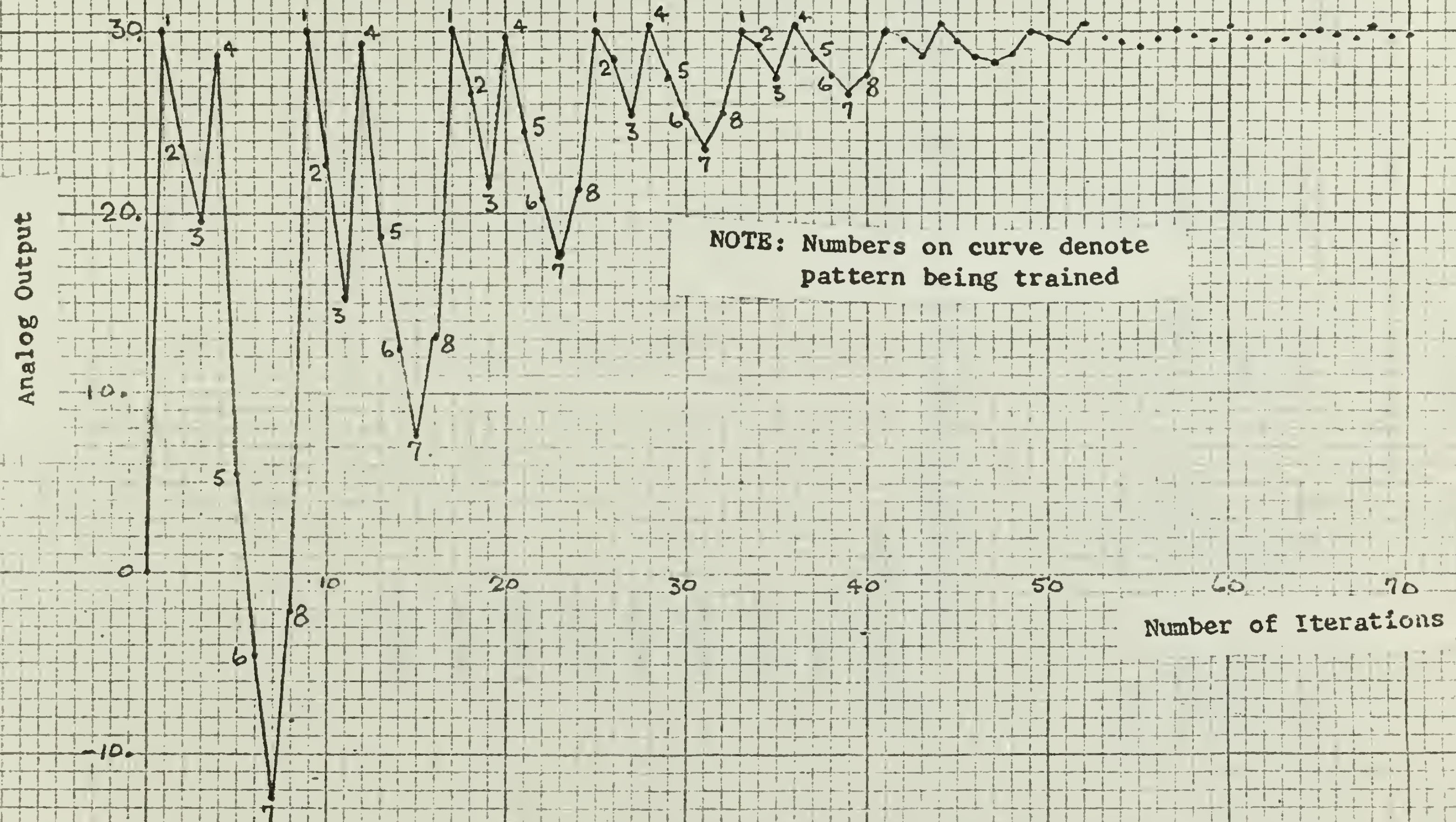
3.540

|  |  |  |  |
|---|---|---|---|
| 7.881 | 10.304 | -5.277 | -5.918 |
| -.730 | 9.141 | .285 | -11.706 |
| -18.106 | -12.534 | -16.846 | -15.344 |
| -.401 | -8.312 | -9.827 | 4.516 |

Analog output of each pattern after convergence.

| Pattern | Analog output |
|---|---|
| 1 | 30.000 |
| 2 | 30.000 |
| 3 | 30.000 |
| 4 | 30.000 |
| 5 | -30.000 |
| 6 | -30.000 |
| 7 | -30.000 |
| 8 | -30.000 |

Note: The criterion of the relaxation adaption procedure is that the magnitude of the analog output for each pattern be equal to or greater than the adaption dead zone. In this particular example the magnitude of each pattern analog output after training was exactly equal to adaption dead zone. This would not normally be expected .



RELAXATION ADAPTION

Modified Relaxation Adaption

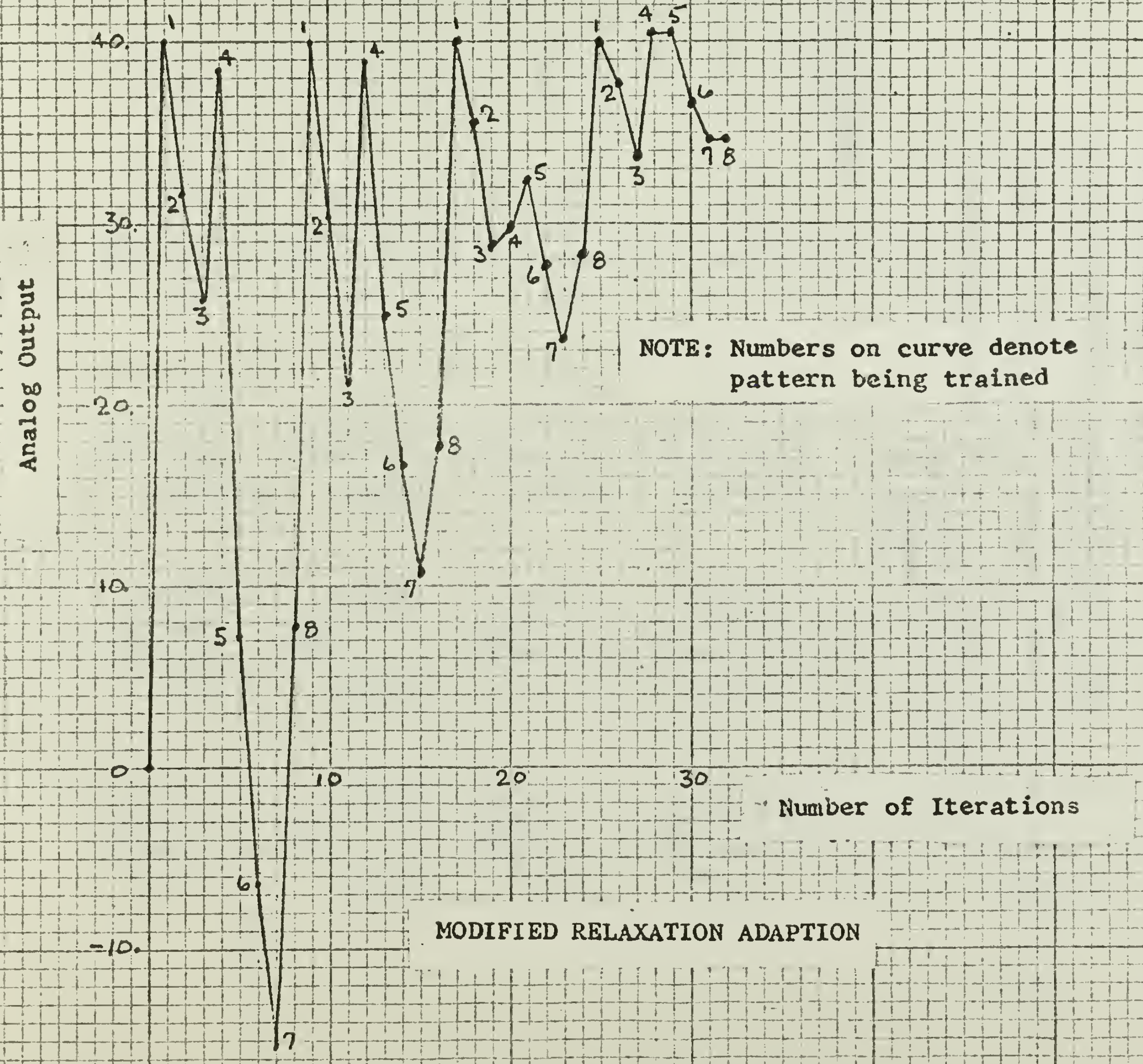
Number of iterations required to converge: 31

Weight elements after convergence.

```
3.896
          9.918     12.819     -5.197     -6.787

         -1.319      9.970       .057    -13.871

        -20.536    -14.237    -20.485    -16.814

         -1.542     -9.604    -11.569      5.735
```

Analog output of each pattern after convergence.

| Pattern | Analog output |
|---|---|
| 1 | 34.697 |
| 2 | 37.435 |
| 3 | 30.138 |
| 4 | 30.844 |
| 5 | -36.076 |
| 6 | -37.091 |
| 7 | -40.000 |
| 8 | -33.365 |



NOTE: Numbers on curve denote pattern being trained

MODIFIED RELAXATION ADAPTION

## APPENDIX III

## MINIMUM SQUARE ERROR ADAPTION AS A FUNCTION OF PROPORTIONAL CONSTANT

An investigation of minimum square error adaption was conducted to determine the effect of the proportional constant. The number of iterations required for convergence and the resultant value of the weight elements were examined. Fixed conditions of the experiment were:

Adaption level: 30.00

Bound for minimum square error: ±1.00

Weights: Continuously variable with initial values of 0.0

Pattern sequences: A. 1, 2, 3, 4, 5, 6, 7, 8
B. 1, 5, 2, 6, 3, 7, 4, 8
C. Random

Training patterns

```
X & J      X . X .     . X . X     . . . .     . . . .
           . X . .     . . X .     X . X .     . X . X
 (+1)      X . X .     . X . X     . X . .     . . X .
           . . . .     . . . .     X . X .     . X . X
             (1)         (2)         (3)         (4)

           . . X .     . . . X     . . . .     . . . .
           X . X .     . X . X     . . X .     . . . X
 (-1)      X X X .     . X X X     X . X .     . X . X
           . . . .     . . . .     X X X .     . X X X
             (5)         (6)         (7)         (8)

C & T      X X X .     . X X X     . . . .     X X X X
           X . . .     . . . X     X . . X     X . . X
 (+1)      X . . .     . . . X     X . . X     X . . X
           X X X .     . X X X     X X X X     . . . .
             (1)         (2)         (3)         (4)

           X X X .     . . . X     . . . .     . . X .
           . X . .     X X X X     X . . .     . . X .
 (-1)      . X . .     . . . X     X X X X     . . X .
           . X . .     . . . .     X . . .     . X X X
             (5)         (6)         (7)         (8)
```

Training patterns

```
X & O       X o X o     o X o X     o o o o     o o o o
            o X o o     o o X o     X o X o     o X o X
 (+1)       X o X o     o X o X     o X o o     o o X o
            o o o o     o o o o     X o X o     o X o X
              (1)         (2)         (3)         (4)

            X X X o     o X X X     o o o o     o o o o
            X o X o     o X o X     X X X o     o X X X
 (-1)       X X X o     o X X X     X o X o     o X o X
            o o o o     o o o o     X X X o     o X X X
              (5)         (6)         (7)         (8)

U & T       X o o X     X X X X     X X X X     X X X X
            X o o X     X o o o     X o o X     o o o X
 (+1)       X o o X     X o o o     X o o X     o o o X
            X X X X     X X X X     X o o X     X X X X
              (1)         (2)         (3)         (4)

            X X X X     o o o X     o X X o     X o o o
            o X X o     X X X X     o X X o     X X X X
 (-1)       o X X o     X X X X     o X X o     X X X X
            o X X o     o o o X     X X X X     X o o o
              (5)         (6)         (7)         (8)
```

NUMBER OF ITERATIONS REQUIRED TO CONVERGE FOR MINIMUM ERROR ADAPTION

| | | PROPORTIONAL CONSTANT | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| PATTERN | SEQUENCE | .25 | .50 | .75 | 1.00 | 1.25 | 1.50 | 1.75 | 1.90 | 2.00 |
| X&J | A | 434 | 188 | 104 | 64 | 42 | 72 | 162 | 455 | * |
| | B | 423 | 182 | 109 | 66 | 49 | 71 | 162 | 434 | * |
| | C | 440 | 200 | 136 | 96 | 72 | 99 | 160 | 392 | * |
| C&T | A | 143 | 68 | 37 | 47 | 35 | 52 | 132 | 353 | * |
| | B | 149 | 68 | 46 | 49 | 41 | 49 | 123 | 343 | * |
| | C | 139 | 72 | 56 | 40 | 72 | 80 | 168 | 383 | * |
| X&O | A | 297 | 129 | 73 | 45 | 45 | 69 | 122 | 360 | * |
| | B | 300 | 144 | 84 | 54 | 47 | 66 | 145 | 367 | * |
| | C | 312 | 152 | 96 | 72 | 72 | 85 | 143 | 366 | * |
| U&T | A | 96 | 55 | 35 | 27 | 37 | 58 | 92 | 259 | * |
| | B | 97 | 51 | 35 | 31 | 45 | 60 | 149 | 397 | * |
| | C | 104 | 56 | 40 | 45 | 40 | 72 | 120 | 280 | * |

* denotes nonconvergence

RESULTANT WEIGHT ELEMENTS AFTER CONVERGENCE FOR X & J PATTERNS

| PROPORTIONAL CONSTANT | PATTERN SEQUENCE | | WEIGHT VALUES AFTER CONVERGENCE | | | |
|---|---|---|---|---|---|---|
| .25 | A | 3.454 | | | | |
| | | | 7.661 | 10.010 | -5.114 | -5.758 |
| | | | -.737 | 8.868 | .265 | -11.380 |
| | | | -17.576 | -12.174 | -16.368 | -14.892 |
| | | | -.425 | -8.074 | -9.559 | 4.389 |
| 1.00 | A | 3.436 | | | | |
| | | | 7.864 | 10.197 | -5.141 | -5.773 |
| | | | -.725 | 9.090 | .228 | -11.528 |
| | | | -17.821 | -12.380 | -16.595 | -15.091 |
| | | | -.400 | -8.237 | -9.718 | 4.442 |
| 1.90 | A | 3.519 | | | | |
| | | | 7.936 | 10.373 | -5.306 | -5.979 |
| | | | -.859 | 9.161 | .202 | -11.658 |
| | | | -18.137 | -12.682 | -16.912 | -15.342 |
| | | | -.448 | -8.136 | -0.811 | 4.694 |
| .25 | B | 3.446 | | | | |
| | | | 7.622 | 9.991 | -5.137 | -5.742 |
| | | | -.713 | 8.869 | .285 | -11.352 |
| | | | -17.575 | -12.162 | -16.328 | -14.895 |
| | | | -.394 | -8.059 | -9.548 | 4.380 |
| 1.00 | B | 3.406 | | | | |
| | | | 7.670 | 10.244 | -5.334 | -5.698 |
| | | | -.663 | 9.199 | .388 | -11.464 |
| | | | -17.932 | -12.542 | -16.403 | -15.285 |
| | | | -.259 | -8.121 | -9.846 | 4.478 |

| PROPORTIONAL CONSTANT | PATTERN SEQUENCE | | WEIGHT VALUES AFTER CONVERGENCE | | | |
|---|---|---|---|---|---|---|
| 1.90 | B | 3.569 | | | | |
| | | | 7.950 | 10.375 | -5.356 | -6.014 |
| | | | -.920 | 9.144 | .220 | -11.739 |
| | | | -18.160 | -12.729 | -16.966 | -15.378 |
| | | | -.418 | -8.154 | -9.782 | 4.650 |
| .25 | C | 3.426 | | | | |
| | | | 7.672 | 10.099 | -5.116 | -5.740 |
| | | | -.720 | 8.878 | .259 | -11.357 |
| | | | -17.573 | -12.172 | -16.367 | -14.878 |
| | | | -.389 | -8.063 | -9.527 | 4.394 |
| 1.00 | C | 3.467 | | | | |
| | | | 7.803 | 10.130 | -5.193 | -5.853 |
| | | | -.674 | 9.104 | .290 | -11.560 |
| | | | -17.826 | -12.455 | -16.525 | -15.247 |
| | | | -.310 | -8.210 | -9.705 | 4.423 |
| 1.90 | C | 3.657 | | | | |
| | | | 7.762 | 10.126 | -5.357 | -5.943 |
| | | | -.775 | 9.026 | .270 | -11.689 |
| | | | -18.095 | -12.358 | -16.831 | -15.239 |
| | | | -.394 | -3.358 | -9.690 | 4.380 |

COMMENTS

1. The proportional constant strongly affects the number of iterations required to converge. A minimum number of iterations was required when the proportional constant was approximately equal to one. The adaption process will not converge if the proportional constant is equal to two.

2. For each pattern pair, the final values of the weight elements were approximately the same regardless of the proportional constant chosen or the sequence of the training patterns. Only a representative sampling of resultant weights are included in the data, but all the results supported the above conclusion.

## APPENDIX IV

### ADALINE RESPONSE TO SIMILAR PATTERNS

An Adaline was trained, until it had converged to a solution, on the ten patterns shown below. After convergence, 50 "similar" patterns were imposed upon the trained Adaline, the "similar" patterns being generated by randomly changing one pattern element of the training patterns. This test was conducted using each of the four adaption procedures in turn.

Training Patterns

+1 Class

```
X o o X o    o X o o X    o o o o o    o o o o o    X o o o X
o X X o o    o o X X o    X o o X o    o X o o X    o X o X o
o X X o o    o o X X o    o X X o o    o o X X o    o o X o o
X o o X o    o X o o X    o X X o o    o o X X o    o X o X o
o o o o o    o o o o o    X o o X o    o X o o X    X o o o X
   (1)          (2)          (3)          (4)          (5)
```

-1 Class

```
o o o X o    o o o o X    o o o o o    o o o o o    o o o o X
o o o X o    o o o o X    o o o X o    o o o o X    o o o o X
X o o X o    o X o o X    o o o X o    o o o o X    X o o o X
X X X X o    o X X X X    X o o X o    o X o o X    X o o o X
o o o o o    o o o o o    X X X X o    o X X X X    X X X X X
   (6)          (7)          (8)          (9)          (10)
```

The fixed conditions of the experiment were:

Weights: Continuously variable with initial values of 0.0

Pattern sequence: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Proportional constant: 1.00

Minimum square error bound: ± 1.00

Adaption dead zone: 30.00

Incremental constant: 1.00

Adaption level: 40.00

| Adaption Procedure | Number of Iterations | Range of Analog Output for Training Patterns | | Range of Analog Output for Similar Patterns | |
|---|---|---|---|---|---|
| | | + | - | + | - |
| Minimum Square Error | 103 | 30.128 | 29.076 | 35.657 | 19.268 |
| | | 29.587 | 30.837 | 6.508 | 38.684 |
| Incremental | 29 | 62.000 | 36.000 | 58.000 | 18.000 |
| | | 36.000 | 52.000 | 18.000 | 64.000 |
| Relaxation | 99 | 42.274 | 30.000 | 43.564 | 17.099 |
| | | 30.000 | 38.701 | 17.171 | 49.652 |
| Modified Relaxation | 19 | 52.642 | 35.249 | 55.708 | 17.367 |
| | | 30.534 | 46.580 | 17.156 | 57.823 |

## COMMENTS

The range of the analog outputs for the similar patterns is dependent upon the final trained values of the weight elements. In this example, one pattern element was changed in the generation of each similar pattern and this had the effect of changing the sign of one corresponding weight element. Thus the largest change, in this case, would occur when the pattern bit corresponding to the largest weight element is changed in sign.

All of the analog outputs of the similar patterns were of the same sign as the desired binary output. However, if similar patterns had been formed by changing more than one pattern element, it is possible that some of the similar analog outputs would be of a different sign than the desired pattern binary output.

# APPENDIX V

## THE RESPONSE OF AN ADALINE TO TRAINING PATTERNS THAT ARE NOT LINEARLY SEPARABLE

Two pattern classes were chosen so that they were not linearly separable. This was accomplished by assigning the same pattern to both the (+1) and the (-1) class. There were a total of 100 input patterns, each of which was composed of nine pattern elements. The training patterns used in this experiment are tabulated in [1].

The experiment consisted of two parts, the first consisting of an attempt to separate the two pattern classes by imposing all the training patterns on the Adaline. This was performed for both 500 and 5000 iterations. Second, a random sample of ten patterns was chosen and Adaline was trained on these patterns for 200 iterations or until it had converged to a solution. Then the remaining 90 patterns were imposed upon the trained Adaline. The fixed conditions for this experiment are the same as those listed in Appendix IV.

The results are listed in the following table. For this experiment, a pattern was defined to be not correctly identified if its analog output was either of an opposite sign to the desired pattern binary output, or if the analog output was zero.

Number of Patterns Not Correctly Identified

| | | Adaption Procedures | | | |
|---|---|---|---|---|---|
| Number of Training Patterns | Number of Iterations | Min. Sq. Error | Inc. | Relaxation | Modified Relaxation |
| 100 | 500 | 13 | 4 | 7 | 7 |
| 100 | 5000 | 13 | 2 | 5 | 5 |
| 10* | 200 | 11 | 9 | 9 | 9 |
| 10* | 200 | 14 | 10 | 10 | 10 |
| 10* | 200 | 14 | 9 | 10 | 9 |
| 10* | 200 | 22 | 11 | 11 | 7 |

* Different random samples of ten patterns.

## APPENDIX VI

## NUMERIC RECOGNITION TESTS

A computer simulated Madaline consisting of four Adalines was employed to attempt the separation of the ten numeric characters into the ten different classes, zero through nine. Ten hand written digits (one set) were obtained from each of five persons, and were converted into patterns in a seven by seven input space. The computer program was written in Fortran and executed on a CDC 1604 digital computer using the Minimum Square Error adaption scheme with a proportionality constant of one and an adaption dead zone of 30. The following output coding scheme was used in this investigation:

| Digit | Adaline 1 | Adaline 2 | Adaline 3 | Adaline 4 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | -1 | -1 | -1 | 1 |
| 2 | 1 | -1 | 1 | -1 |
| 3 | 1 | 1 | 1 | -1 |
| 4 | -1 | -1 | 1 | 1 |
| 5 | 1 | 1 | -1 | -1 |
| 6 | -1 | 1 | 1 | 1 |
| 7 | 1 | -1 | -1 | -1 |
| 8 | 1 | 1 | -1 | 1 |
| 9 | -1 | 1 | -1 | 1 |

A series of tests were run in which one, two or more sets of patterns were used to train the Madaline, after which the Madaline was asked to recognize and classify all fifty input patterns. The number of training iterations performed prior to the classification check, and the number of digits recognized out of the total of fifty imposed on the system are included in the data tables below.

| No. of patterns trained on | No. of training iterations | No. of patterns checked | No. of patterns classified |
|---|---|---|---|
| 1 set of 10 | 4 iterations per pattern | 50 | 36 |
| 2 sets of 10 | 4 iterations per pattern | 50 | 36 |
| 3 sets of 10 | 4 iterations per pattern | 50 | 44 |
| 4 sets of 10 | 4 iterations per pattern | 50 | 45 |
| 5 sets of 10 | 4 iterations per pattern | 50 | 50 |

| No. of patterns trained on | No. of training iterations | No. of patterns checked | No. of patterns classified |
|---|---|---|---|
| 1 set of 10 | 200 total iterations | 50 | 37 |
| 2 sets of 10 | 200 total iterations | 50 | 38 |
| 3 sets of 10 | 200 total iterations | 50 | 44 |
| 4 sets of 10 | 200 total iterations | 50 | 44 |
| 5 sets of 10 | 200 total iterations | 50 | 50 |

The results of this test show that the system must be trained on all patterns to insure that it will be capable of classifying all the patterns. However, training on just one set gave the system the capability of recognizing many of the other patterns submitted to it for classification. Of interest is the suggestion that repeated iterations do not necessarily improve the ability of the system to classify the imposed inputs.

## APPENDIX VII

## FORTRAN SYMBOL CLASSIFICATION

The sole purpose of this test was to determine whether or not 45 hand printed Fortran symbols could be separated by a computer simulated Madaline consisting of the minimum number of Adalines that could theoretically accomplish this task. Six Adalines, each with a seven by seven input space, were utilized to accomplish the desired separation after the hand written characters had been converted into input patterns. The computer program was written in Fortran and executed on a CDC 1604 digital computer using the Minimum Square Error adaption technique. A proportionality constant of one and an adaption dead zone of 30 were used for this test.

After extensive manipulation of the Madaline output coding schemes, the 45 characters were properly classified. Nine thousand training iterations were executed prior to the classification of the 45 input patterns. The 45 Fortran symbols and the output coding scheme which accomplished the desired classification are as follows:

| Pattern | Adaline One | Adaline Two | Adaline Three | Adaline Four | Adaline Five | Adaline Six |
|---|---|---|---|---|---|---|
| 0 | -1 | 1 | 1 | 1 | -1 | 1 |
| 1 | 1 | 1 | 1 | 1 | -1 | 1 |
| 2 | -1 | 1 | -1 | 1 | -1 | 1 |
| 3 | -1 | 1 | -1 | -1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | -1 |
| 5 | 1 | 1 | 1 | -1 | -1 | -1 |
| 6 | -1 | 1 | -1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | -1 | -1 | 1 |

| Pattern | Adaline One | Adaline Two | Adaline Three | Adaline Four | Adaline Five | Adaline Six |
|---|---|---|---|---|---|---|
| 8 | 1 | 1 | -1 | 1 | 1 | -1 |
| 9 | 1 | 1 | -1 | -1 | 1 | 1 |
| A | -1 | 1 | 1 | -1 | 1 | 1 |
| B | 1 | -1 | 1 | 1 | 1 | 1 |
| C | -1 | 1 | 1 | -1 | -1 | 1 |
| D | 1 | -1 | 1 | 1 | -1 | 1 |
| E | 1 | -1 | 1 | 1 | 1 | -1 |
| F | 1 | -1 | 1 | -1 | 1 | -1 |
| G | -1 | 1 | -1 | 1 | -1 | -1 |
| H | 1 | -1 | 1 | -1 | -1 | -1 |
| I | -1 | 1 | 1 | 1 | 1 | 1 |
| J | 1 | 1 | -1 | -1 | 1 | 1 |
| K | 1 | -1 | 1 | 1 | -1 | -1 |
| L | 1 | -1 | 1 | -1 | -1 | 1 |
| M | 1 | 1 | 1 | 1 | -1 | -1 |
| N | 1 | -1 | -1 | 1 | -1 | 1 |
| O | 1 | 1 | 1 | 1 | 1 | 1 |
| P | 1 | -1 | 1 | -1 | 1 | 1 |
| Q | -1 | 1 | -1 | 1 | 1 | -1 |
| R | 1 | 1 | 1 | -1 | 1 | -1 |
| S | -1 | 1 | 1 | 1 | 1 | -1 |
| T | 1 | 1 | 1 | -1 | 1 | 1 |
| U | 1 | -1 | -1 | 1 | 1 | -1 |
| V | -1 | 1 | -1 | -1 | -1 | -1 |
| W | -1 | 1 | 1 | -1 | 1 | -1 |
| X | -1 | 1 | 1 | -1 | -1 | -1 |

| Pattern | Adaline One | Adaline Two | Adaline Three | Adaline Four | Adaline Five | Adaline Six |
|---|---|---|---|---|---|---|
| Y | -1 | 1 | 1 | 1 | -1 | -1 |
| Z | -1 | -1 | 1 | -1 | -1 | -1 |
| + | 1 | 1 | -1 | 1 | 1 | 1 |
| - | -1 | -1 | 1 | -1 | 1 | 1 |
| / | -1 | -1 | 1 | 1 | -1 | -1 |
| ( | -1 | -1 | 1 | -1 | 1 | -1 |
| ) | -1 | -1 | 1 | 1 | -1 | 1 |
| , | -1 | -1 | -1 | -1 | 1 | -1 |
| . | -1 | -1 | -1 | -1 | -1 | 1 |
| = | -1 | -1 | -1 | 1 | -1 | 1 |
| * | -1 | -1 | -1 | 1 | 1 | 1 |

The final output coding scheme was obtained mainly by trial and error methods. However, an effort was made to use certain characteristics of the individual input patterns to facilitate the desired classification. Specifically, at least one Adaline was trained to produce the same output for each of the following sets of input patterns:

| | | |
|---|---|---|
| a) | Patterns with a long vertical line in the left hand column. | Example: E,L,P |
| b) | Patterns with other long vertical lines. | Example: 4,I,T |
| c) | Patterns with a long horizontal line. | Example: E,H,I |
| d) | Patterns with large circles | Example: O,Q |
| e) | Patterns with small circles | Example: 8,9,P |
| f) | Patterns with small horizontal lines | Example: A,+,- |
| g) | Patterns with small vertical lines | Example: 5,+ |
| h) | Patterns with left to right slant lines | Example: N,V,X |

i) Patterns with right to left slant lines Example: K, Z, /

j) Patterns which were smaller than the others Example: °, °

[redacted]